IMAGE INPAINTING

Major project report submitted in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

in

Electronics and Communication Engineering

Submitted by

A Y S HARSHA

(16B81A04P9)

SRI HARSHA SAGAR

(16B81A04L0)

D SOWMYA NAIDU

(16B81A04K3)

Department of Electronics & Communication Engineering

CVR COLLEGE OF ENGINEERING

(An Autonomous Institution & Affiliated to JNTUH)

Ibrahimpatnam (M), Ranga Reddy (D), Telangana

2016-2020

IMAGE INPAINTING

Major project report submitted in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

in

Electronics and Communication Engineering

Submitted by

A Y S HARSHA

(16B81A04P9)

SRI HARSHA SAGAR

(16B81A04L0)

D SOWMYA NAIDU

(16B81A04K3)

Under the Supervision of

Mr.B.Janardhana Rao

Associate Professor

Department of Electronics & Communication Engineering

CVR COLLEGE OF ENGINEERING

(An Autonomous Institution & Affiliated to JNTUH)

Ibrahimpatnam (M), Ranga Reddy (D), Telangana

2016-2020

Cherabuddi Education Society's
CVR COLLEGE OF ENGINEERING
(An Autonomous Institution)
ACCREDITED BY NATIONAL BOARD OF ACCREDITATION, AICTE
(Approved by AICTE & Govt. of Telangana and Affiliated to JNT University)
Vastunagar, Mangalpalli (V), Ibrahimpatan (M), R.R. District, PIN - 501 510
Web : http://cvr.ac.in, email : info@cvr.ac.in
Ph : 08414 - 252222, 252369, Office Telefax : 252396, Principal : 252396 (O)

CERTIFICATE

This is to certify that the project titled "**IMAGE INPAINTING"** submitted to the **CVR College of Engineering,** affiliated to **JNTU, Hyderabad** by **A Y S Harsha (16B81A04P9), Sri Harsha Sagar (16B81A04L0), D. Sowmya Naidu (16B81A04K3)** bonafide record of the work done by the students towards partial fulfillment of requirements for the award of the degree of **Bachelor of Technology in Electronics & Communication Engineering**.

Supervisor **Head of the Department**

Mr. B Janardhana Rao
Associate Professor
Department of Electronics &
Communication Engineering

Dr. K. Lalithendra
Professor & HOD
Department of Electronics &
Communication Engineering

Place: Hyderabad

Date:

ACKNOWLEDGEMENT

Completion of this project and thesis would not have been possible without the help of many people, to whom we are very thankful. First of all, we would like to convey our sincere thanks to **Dr. K. S. Nayanathara**, **Principal**, CVR College of Engineering for her constant support and encouragement.

We would like to express my sincere gratitude to our supervisor, **B.Janardhana Rao, Associate professor.** His constant motivation, guidance and support helped us a great deal to achieve this feat.

We would like to thank **Dr. K. Lalithendra, Professor of ECE & HOD**, for guiding and inspiring us in many ways. We are also thankful to other faculty members and staff of Electronics and Communication department for their support.

We also thank our coordinator **Dr**.V.**Arthi, Associate professor** for his/her constant support in guiding us about the academic related work.

We wish a deep sense of gratitude and heartfelt thanks to management for providing excellent lab facilities and tools. Finally, we thank our seniors whose guidance helped us in this regard.

ABSTRACT

The image inpainting quality depends on the method of patch priority calculation and best matching patch selection method. Image inpainting can be applied in repairing the damaged images and removing the unwanted objects in the images. With reference to all the existing methods of image inpainting, the patch priority calculation and best matching patch selection play a major role in getting good results. In this paper, an improved patch priority calculation method is proposed by introducing regularization factor μ and adaptive coefficients. The best matching patch selection is determined using two patch searching methods such as the sum of squared error (SSE) and the sum of absolute difference (SAD) distance methods. The best combination is identified with respect to μ and adaptive coefficients values for highest patch priority calculation. This best combination is used along with SSE and SAD patch selection method for repairing the damaged images and removing of the object in images. The results from the proposed inpainting method are compared with the existing inpainting methods. The metrics of proposed inpainting, such as peak signal to noise ratio, mean square error and time elapsed for different images are obtained and compared. The results for the several images obtained from the proposed inpainting method using MATLAB.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The restoration and modification of images in a way that is not detectable by an observer who has not seen the original picture is a very old practice – dating back to the manual restoration of medieval art by filling in any gaps that may have distorted the artwork over the years. In modern times, the digitizing of analog images to 'live forever' and preventing their decay, often results in defects like scratches, etc. that have to be removed. This practice is known as inpainting. The aim of inpainting is to reconstruct the missing or damaged portions of the picture, in order to restore its unity. The obvious need to restore images extends from paintings to photographs and films. The purpose, however, remains the same – to recondition any deterioration (e.g., cracks or scratches in paintings and photographs), or to modify (e.g., add or remove elements like red eyes), the goal being to produce a modified image in which the inpainted region is merged into the original image so well that an observer is not aware of the modification. The filling-in of missing information is an important aspect of image processing, with applications that range from image restoration, to image coding and transmission (e.g., recovering lost packets), and special effects (e.g., removal of objects).

Traditionally, artists performed image inpainting manually which was a very cumbersome and tedious process. Motivated in part by the work of Nitzberg-Mumford and Masnou-Morel, Bertalmio et al have developed algorithms for digital inpainting of still images that produce very impressive results [1]. However, the algorithm usually takes several minutes for the inpainting of a relatively small area, prompting our group to research other faster algorithms that can produce similar results in lesser time, with added features

1.1 Image Inpainting

Image inpainting technology aims to fill the missing region in an image. According to the inpainting theory, it can be divided into two categories, diffusion-based and exemplar-based inpainting methods. Diffusion-based methods inpaint images on pixel-level, while exemplarbased methods inpaint images on patch-level. Image inpainting technology is used for repairing the images destroyed, but now it is extended widely to apply on object removal, image restoration, image compressing and other applications.

In the existing exemplar-based image inpainting algorithms, the Sum of Squared Differences (SSD) method is employed to measure the similarities between patches in a fixed size, and then using the most similar one to inpaint the destroyed region. However, sometimes

only calculating the SSD difference would produce a discontinuous structure and blur the texture. To solve this problem, we firstly optimize the inpainting priority function and proposed an adaptive patch method to obtain more significant patches. The adaptive patch method changes the size of the patch by computing the patch sparsity. Secondly the proposed method calculates the maximum similarity between patches in different rotation angles so that it obtains the most similar rotation invariant matching patch. From the experimental results, the proposed method can improve the accuracy of the patch selection process compared with the traditional methods, and the proposed method can keep a better global visual appearance, especially for the image which contains more structure contents and the images whose destroyed region has a large width.

1.2 History

The most fundamental inpainting approaches is the diffusion-based inpainting algorithm. Bertalmio firstly introduces the theory of image inpainting and gives this idea in 2000. Bertalmio proposes a model based on isophote to continuously propagate the undamaged information into the destroyed region. Chan and Shen proposed a variational model based on total variation (TV) to repair the destroyed information. And they overcome the deficiency in the TV model that it cannot realize the connectivity information by proposing a curvature-driven diffusion equation. Image inpainting based on fast marching method (FMM) becomes a research hotspot after Telea' method. FMM expands the traditional isophote model into three items (the direction, the distance, and the isophote influences) to compute the pixel's gray value. The advantage of FMM in calculation speed and in inpainting large destroyed region influences other researchers. Yang introduces the anisotropic diffusion theory into the FMM and uses it to improve the visual effect for inpainting the image which has a large region. These methods mainly focuses on inpainting the missing areas on pixellevel and they performs a good global effect for the image with small width destroyed region. Another advantage of diffusion-based approach is the fast calculation speed. Ullo employs the Complex Ginzburg-Landau equation to improve the contrast and applies it on restoring the SAR interferograms. Hu proposes a method based on TV model to restore the image generated from the lateral multi-lens video logging device. Andris applies inpainting method on image compressing field, the method develops a proof-of-concept codec which combines inpainting technology based on partial differential equations with the variational optic flow model.

In this paper, the image inpainting method is proposed using the sum of squared error (SSE) and the sum of absolute difference (SAD) methods to find out the best matching patch selection, and a fewnewfactors are introduced for the determination of patch priority to be filled first in the target region. Section 3 describes the basicexemplar-based inpainting method. Section 4 presents the proposed inpainting algorithm using patch priority calculation method and best matching patch selection method. The experimental results for several images are presented in Section 6 and compared with existingmethods. The existingmethods, Criminisi and JingWang, are considered for the comparison of object removal results. The Criminisi's Bhattacharya distance (CBD) method and structure consistency inpainting methods are considered for the comparison of repairing damaged image results. The CBD, a weighted Bhattacharya distance method, is used to solve the dropping effect problem. In this method, overall intensity distribution values are considered but not the spatial distribution.The structure consistency method uses Fast Fourier Transform for patch matching by taking the distribution of source and target patch differences.

2. LITERATURE SURVEY

In 2006, Criminisi et al. [2] proposes an approach that inpaint the image not on pixel level but on patch level, we also call it as exemplar-based inpainting algorithm. It selects the most similar patch to adjust the destroyed region. In this method, firstly it decides the inpaint order by checking the known pixels in the neighbourhood of every destroyed edge pixels, then calculating the difference and selecting the most similar patch from the known region, at last uses the patch to inpaint the destroyed region. Under this model, by modifying the inpaint priority and the matching method, other researchers improve the patch inpainting method and perfect this method. In general, exemplar-based inpainting methods keep the image texture and improve the effect even the inpainting region is large.

Ram [3] focuses on the order of the inpainting, by reordering the patches it can obtain a good recovery of the image. In, a novel scheme is proposed to use both low-solution and high-solution information to get similar patches. Also a lot of researchers try to use structure information to matching the similar patches.

Huang [4] proposes a method to use structure consistency obtaining the most similar patch. Discrete cosine transform is also employed to calculate the similarities between patches,

Ou [5] uses it to select more significant patch. In the past years dictionary learning technology has become more and more mature, so that image inpainting researchers also employed it to optimize the inpainting method and improve the inpainting effect. It can be regard as an approach of exemplar-based inpainting algorithm, exemplar-based inpainting algorithm aims at finding the most similar patch to inpaint the destroyed patch and dictionary learning algorithm tries its best to generate a new patch to inpaint the destroyed one.

Elad [6] proposes a sparse representation-based inpainting method. In this paper a dictionary learning method is used for creating a dictionary for sparse representations, via a singular value decomposition (SVD) approach. This method inpaints the images by iteratively alternating between sparse coding the image patches based on the current dictionary, and updating the atoms in the dictionary to better fit the coefficients. In this method Elad also employs orthogonal matching pursuit (OMP) algorithm to calculate the coefficients. This structure also affects a lot of researchers.

Xu [7] combines exemplar-based inpainting methods with dictionary learning-based inpainting methods after analysis the advantages and disadvantages of them. Xu uses ten most similar patches in the source image as the atoms of the dictionary. And employing the locally

linear embedding (LLE) theory to obtain a linear combination of the candidate patches, finally this method generates the new patch.

Ruzic [8] builds the dictionary by dividing the image into variable size blocks according to their context. So not only the patch information, but also their context information were used to improve the inpainting result.

Papyan [9] improves the patch log likelihood algorithm by considering a multi-scale prior. This method calculates the priority of the patches in different scales and it can provide a smoother priority in inpainting. With the help of dictionary learning technology, there also comes out a lot of applications.

Wang [10] proposes a novel inpainting model carried on the dictionary learning by method of directions (MOD), and using it to repair missing region of murals in Potala Palace.

Munawar [11] employs the higher-order Boltzmann machine and applies it on detecting anomalies on the road. This method tries to inpaint the road patches by training the commonly occurring road features such as lane markings and expansion dividers, depending on the context. Applies inpainting technology on depth map reconstruction and proposes a Kinect-based underwater depth map estimation method.

In Knutsson and Westin [12] the problem of the image analysis when performed on irregularly sampled image data is considered under the theory of signal and its certainty. This is to consider the separation of both data and operator applied to the data in a signal part and a 'certainty' part. Missing data in irregularly sampled series is handled by setting the certainty of the data equal to zero. In the case of uncertain data, an estimate of certainty accompanies the data, and this can be used in a probabilistic framework. The theory that they developed following these ideas is called Normalized Convolution.

Another convolution based technique by Oliveira [13], repeatedly convolves a filter over the missing regions so that the information from the edges is diffused inwards to the corrupted region. Both these convolution processes are reasonable in reconstructing images but falter in terms of resolution. The corrupted segments get populated, but everything tends to get blurred.

3. ALGORITHMS

The different types of algorithms, in the past, have broadly been classified as: (i) Texture Synthesis algorithms that generate image regions from sample textures, and (ii) Inpainting techniques that fill in small gaps and holes in the pictures. Texture Synthesis algorithms work better with 'textures', two-dimensional patterns that repeat; Inpainting techniques focus on linear structures such as lines and contours that can be thought of as one-dimensional patterns.

The convolution method exhibited compelling reasons to select it. It was the fastest algorithm on thin defects (less than 9 pixels across) producing results comparable to the other candidates.

Traditional texture synthesis models, as well as variational models were considered. They both inpainted large sized defections. However, the variational models respected geometrical properties of the picture. This was done by calculating a set of partial differential equations to determine how to grow the isophotes (bands of equal color and intensity). However, as diffusion was part of the process, blurring would occur as the inpainting proceeded inwards. Research has shown that it is very computationally expensive with the inpainting time proportional to the area of the defect.

Traditional methods of texture synthesis would grow inward from the boundary of the corrupted region. Resolution is maintained as there is no diffusion involved – textures sampled from the rest of the image are plugged into the inpainted block.

However, both traditional texture synthesis and the variational models were discarded as most images are a combination of textures and geometry. The chosen algorithm, the exemplar based algorithm, was chosen. It maintains the resolution of a picture, using texture synthesis methods. However, the synthesis is guided by geometrical attributes in an image (the colors representing different objects in a picture). It finds and linearly extends the isophotes of the image through texture synthesis, growing these sharp color gradients before attacking the rest of the boundary, maintaining the lines and edges that define the geometrical attributes of an image. It thus combines the advantages of the variational and texture synthesis methods, leaving behind their shortcomings. However, it is more computationally expensive than the convolution method. As traditional texture synthesis methods don't find color gradients to grow on, this method is more expensive than traditional texture synthesis algorithms.

3.1 Convolution Method

The technique of blurring the colors out into the missing areas is called Convolution. Mathematically, repeated blurring and diffusion are identical. Isotropic diffusion is the idea behind these methods. When an image is blurred, the colors of each pixel are averaged with a small portion of the color from neighboring pixels. That pixel, in turn, contributes a small part of its color to each of its neighbors, that is, colors from the untainted areas are spread into the corrupted zones in an even way. Directions and isotopes are not given a weighting and as the zones of corruption are narrow, we don't have to diffuse very far and the deformations resulting from such diffusion are not very great.

This method works well for deformations that are not big. A large amount of surface area may be corrupted, but as long as the corruptions themselves are 'thin' in nature, the algorithm works well. This is good as most deformations are pen marks and scratches which are thin in nature. It is also good at filling in the picture when we only have a few components of the original signal. This happens if you're receiving an image over a noisy or faulty line.

The general idea is to create a kernel of some sort (an N x N matrix) that is repeatedly convolved over areas of picture, which fills in a pixel based on surrounding pixel values. The values in the matrix determine the way it will spread colors from the surrounding scene into the corrupted zone. This process is repeated over and over until the image is restored. Such a method has the benefit of being very fast as only multiplication is used. An example of such a kernel is given below:

a	b	a
b	0	b
a	b	a

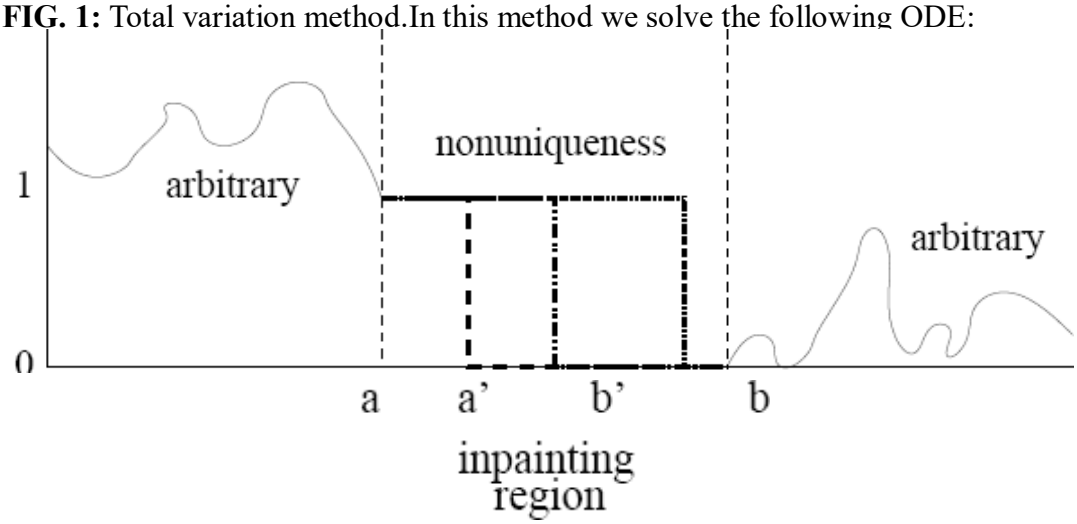
c	c	c
c	0	c
c	c	c

a = 0.073235
b = 0.176765
c = 0.125

3.2 Total Variation Method

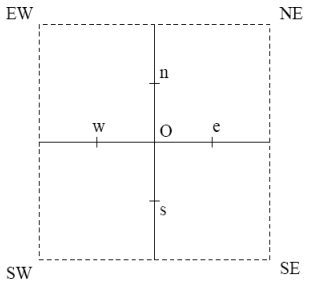
The Total Variation (TV) method minimizes the absolute value of the gradient of the image. The reason for implementing the TV method is because it is much better at dealing with sharp edges than other methods are. For TV there is a high degree of nonuniqueness which holds true even when we minimize the TV over an appropriate space. This lack of uniqueness is because of the high degree of symmetry or the fact that all monotone jumps are treated equally. The numerical solution propagates the boundary data inwards at equal speeds, so a minimizer that is symmetric at the mid-point will be chosen.

FIG. 1: Total variation method.In this method we solve the following ODE:



$$\frac{d}{dx}\left(\frac{u'}{|u'|}\right) = 0 \quad for x \in [a,b] \qquad u(a) = 1,\ u(b) = 0$$

The following graph shows O as the inpainted region and the directions as the computed neighbors. This method uses the neighbors of the corrupted region to calculate the new inpainted region. Each direction (North, South, East, and West) is equated using a circular shift and set as neighbor values (u_E,u_N, etc)



After these are computed, the weights are computed. This computation is shown in the following equation, A. This equation is the gut of the numerical implementation:
The key to this method is in this approximation and the degeneracy of the PDE. After computing the weights, the method calls for approximating the new inpainted region. The equation becomes

$$\begin{aligned} w_P &= \frac{1}{|\nabla u_p|} \quad P \in \Lambda_O \\ 0 = \sum_{P \in \Lambda_O} h_P &= \frac{w_P}{\sum_{Q \in \Lambda_O} w_Q} \end{aligned}$$

by using We can solve this using a Gauss-Jacobi iteration scheme for linear systems, which gives the following update of

$\mathbf{u}^{(n-1)}$ to $\mathbf{u}^{(n)}$ by the following equation:

$$u_O^{(n)} = \sum_{P \in \Lambda_O} h_P^{(n-1)} u_P^{(n-1)}$$

Due to h being a low-pass filter, the iterative algorithm is stable and satisfies the maximum principle. The final process is multiple iterations. This is done to achieve the best result.

3.3 Exemplar Method

The Exemplar method can essentially replicate both texture and structure. However, the success of the method depends highly on the order in which the filling proceeds. The procedure involves selecting a target region to be filled. The size of the template window is specified as a 9x9 pixel window. After the target region and window size have been decided on, we calculate the *confidence term* and the *data term*.

Each pixel, in the algorithm, has a specific colour value, and a confidence value that represents the confidence in the colour value. Furthermore, patches along the fill front have a priority value which establishes the order in which the patches are filled. The general structure of the algorithm is as follows –

Say we start with a patch, Ψ_{p}, centered at point 'p'. The priority of

$$P(\mathbf{p}) = C(\mathbf{p})D(\mathbf{p}).$$

this patch is calculated as: Where C(p) is the confidence term and D(p) is the data term, calculated as:

$$C(\mathbf{p}) = \frac{\sum_{\mathbf{q} \in \Psi_{\mathbf{p}} \cap \bar{\Omega}} C(\mathbf{q})}{|\Psi_{\mathbf{p}}|}, \qquad D(\mathbf{p}) = \frac{|\nabla I_{\mathbf{p}}^{\perp} \cdot \mathbf{n}_{\mathbf{p}}|}{\alpha}$$

Here, $|\Psi_{\mathrm{p}}|$ defined as the area of the patch, and α is the normalization factor. We calculate the priorities for all patches along the boundary of the fill region. The confidence term, C(p), helps in filling those patches first which have more of their pixels filled in already, either because they were never part of the target region, or because they were filled in earlier. The data term D(p), gives patches that have higher colour gradients a higher priority. Thus the priority with which we fill a patch depends on how many pixels are already filled in within that region, and how high a colour gradient exists within that patch.

A representation of the process is explained as –

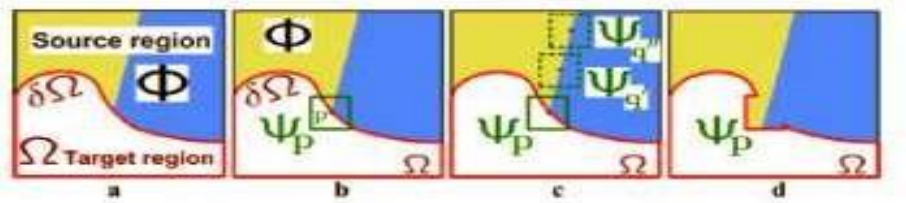


Fig. 2: Exemplar Method

As the filling process goes on, the pixels towards the target boundary or contour will have higher confidence values, and will be filled before the pixels at the center of the target region which have lower confidence values. The Data term D(p) depends on the isophotes hitting the contour on each iteration. A clearer way to understand this procedure is to look at the calculations in the MATLAB code.

The gradient, Ix Iy is calculated for the image coordinates in each of the colors – R G B (Red, blue and green). The gradient measures the change in the color components, thus measuring the change in amount of Red, Blue or Green in each pixel. This means that there would be a higher gradient value at color boundaries. (For example, for the pixels at the boundary of the sky and grass, the gradient value is higher than the gradient values for pixels in the sky region.) This is for Red. The same line is repeated for img(:,:,2) and img(:,:,3) (blue and green)

```
[Ix(:,:,1) Iy(:,:,1)] = gradient(img(:,:,1));
% add Red blue green components, divide by the size Ix =
sum(Ix,3)/(3*255); Iy = sum(Iy,3)/(3*255);
```

Convole the flll region with the patch and for all values>0, store them in dR

```
dR = find(conv2(fillRegion,[1,1,1;1,-8,1 ;1,1,1] ,'same')>0);
```

N stores the gradient of the not fill region. We get the corresponding location in dR from this gradient, and normalizeit.

```
Nx,Ny] = gradient(~fillRegion);
N = [Nx(dR(:)) Ny(dR(:))];
N = normr(N);
```

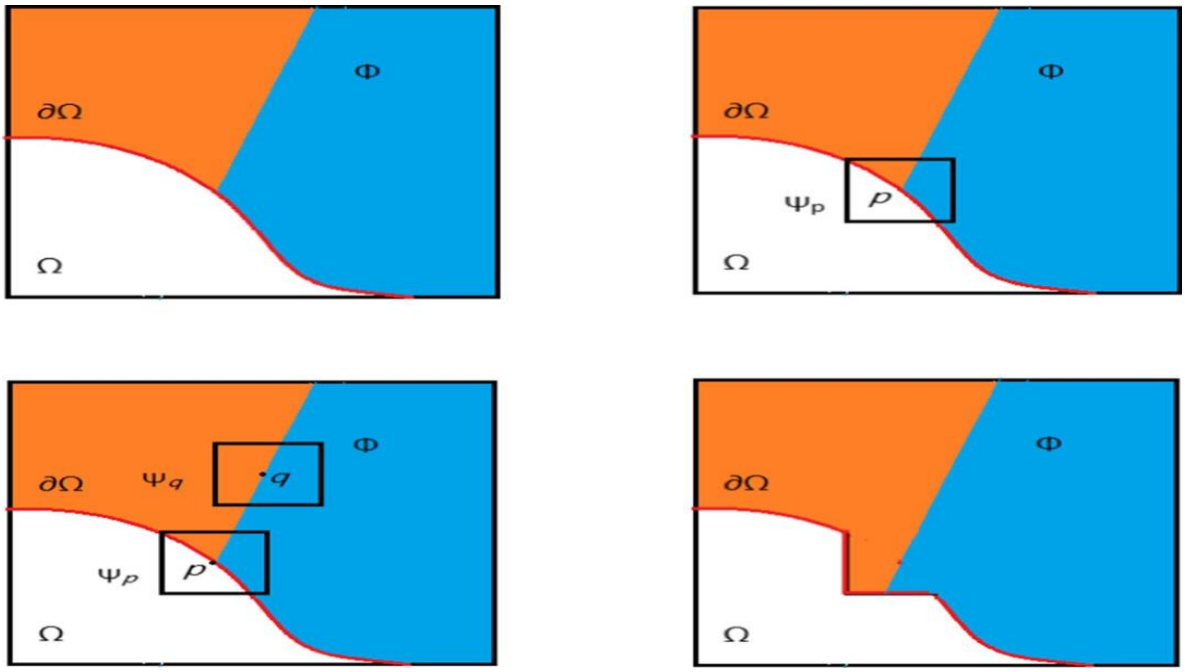
The calculated dR is used to get the patch with the highest priority, say best_patch and fill it with data extracted from the sourceregion.
The priorities and confidence values are calculated as –

```
q = best_patch(~(fillRegion(best_patch)));
C(k) = sum(C(q))/numel(best_patch);
```

Number of elements in best_patch (This code follows the definition for confidence terms as statedearlier)
Patch priorities = confidence term * data term

```
D(dR) = abs(Ix(dR).*N(:,1)+Iy(dR).*N(:,2)) + 0.001;
priorities = C(dR).* D(dR);
```

After this, we find the best exemplar patch, such that it has a minimum difference with respect to the patch being filled in. The data from the best exemplar patch is copied to the patch being filled in, and the confidence values are updated. This procedure is repeated till all the pixels are filled.



4. PROPOSED WORK

In the proposed work, two criteria are considered to improve the inpainting process: (a) the improved patch priority calculation on the boundary of the target region;
(b) the improved best patch selection from the sourceregion to fill the target region.

4.1 IMROVED PATCH PRIORITY METHOD

The existing exemplar-based inpainting fills the target region with best exemplar patches from the source region. Here, the texture as well as structure information of the image is considered to inpaint. This inpainting method suffers from the problem of dropping effect. The dropping effect means the priority drops to a low value due to the rapid decrease in confidence term at less number of iterations. This dropping Effect may be reduced by an improved priority calculation method using regularization factor μ. Then the priority of the target patch can be obtained from the below equation:

$$\boldsymbol{P(p) = a * GC(p) + b * D(p),}$$

where a is a coefficient related to confidence term and b is another coefficient related to data term ranging $a \geq 0$, $b \leq 1$, $a + b = 1$ and the modified confidence term is given by $GC(p)$ below:

$$\boldsymbol{GC(p) = (1 - \mu)C(p) + \mu.}$$

The regularization factor$\mu$is used to control the smoothness of the confidence term curve. The priority of the target patch is determined using texture consistency in the confident term of the target patch surrounding that pixel p. The texture information of the image is consistent in different areas of the image and inconsistent in the same area in the image, which is identified with the help of different regularization factor μ values over the range from0.1 to 0.7 based on exemplar based image inpainting. The highest priority patch is identified by checking different combinations of μ with different combinations of a and b.

4.2 Improved patch selection method

The best matching patch or exemplar patch from the source region to fill the target patch is selected by using two searchingmethods independently,which are SSE and SAD. The patches with minimum distance value using SSE are identified as bestmatching patches. The best patch can be determined by the following equation:

$$\psi_{q^1} = \arg \min_{\psi_a \in \Phi} d_{\mathrm{SSE}}(\psi_p, \psi_q),$$

where dSSE is the distance using the SSE method, as obtained by

$$d_{\mathrm{SSE}}(\psi_p, \psi_q) = \sum (\psi_p - \psi_q)^2,$$

where ψp and ψq target region patch and source region patch, respectively. Another new method SAD is used to find out the best matching patch with respect to source region patches. The minimum distance between the target patch and the patches on the source region using the SAD is given as

$$\psi_{q^1} = \arg \min_{\psi_q \in \Phi} d_{\mathrm{SAD}}(\psi_p, \psi_q),$$

where dSAD is a distance using the SAD method, which is given as,

$$d_{\mathrm{SAD}}(\psi_p, \psi_q) = \sum |\psi_p - \psi_q|.$$

Algorithm 1 explains the step by step process involved in the proposed inpainting method using SSE and SAD independently.

Algorithm 1:

1. Read the Image;

2. Create target region and source region ();

3. Determine the boundary of the target region i.e. ∂;

4. Create patches on the boundary making pixels on the boundary as the centre point of the patch ψp;

5. Obtain the highest priority patch to be filled first i.e. ψpi (i = 1, 2, 3 ...) using Equation(5)Check with different combinations of μ, a and b values;

6. Create sample patches on the source region ψqi (i = 1, 2, 3 ...). The best matching patch is decided by two searching methods: SSE and SAD;

7. Implement SSE and SAD methods for different combinations of μ, a and b values;

8. Consider the best matching patch from the source region to highest priority patch in the target region: i.e. ψq1 to ψp;

9. Update C(p) for the new target region boundary.

5. ANALYSIS AND EXPERIMENTAL RESULTS

In this section, various experimental results with different μ, a and b values for patch priority calculation are presented. The best exemplar patch selection methods such as SSE and SAD are also explained. From all these experiments, the best combination gives a good inpainted image in both the SSE and SAD methods for different μ,a and b values is identified.To check the effectiveness of the best combination the results are compared with the existing methods.

5.1: Removing the object using inpainting

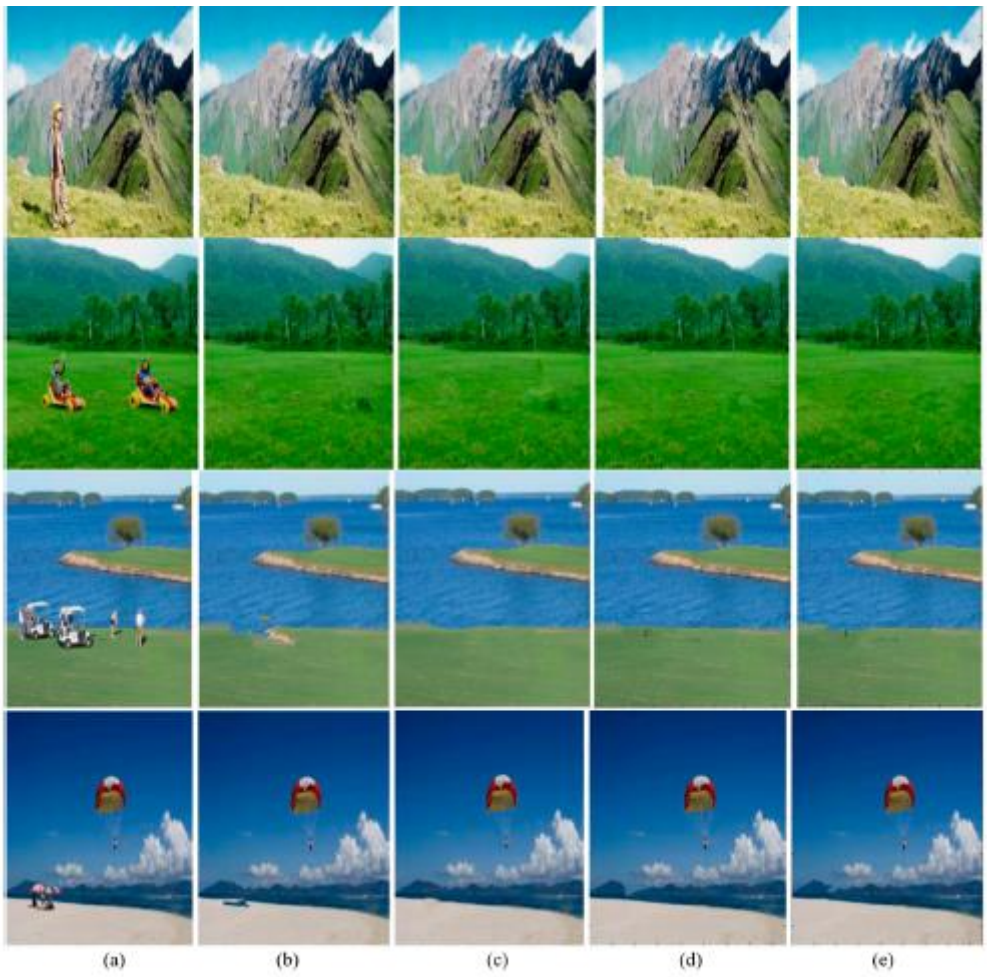


Fig 5.1 Comparison with existing image inpainting methods for small area removal: column (a) Input image; column (b) Results with Criminisi ; column (c) Results with exemplar based image inpainting; column (d) Results with proposed method for μ =0.7, a = 0.8 and b = 0.2 using the SSE method; (e) Results with proposed method for μ = 0.5, a = 0.8 and b = 0.2 using the SAD method.

Table 1: PSNR, MSE and time elapsed for SSE with various μ, a and b values for removing the object

		PSNR (dB)	MSE	Time elapsed (s)
$\mu = 0.7$	$a = 0.2$ $b = 0.8$	19.817	683.873	29.297
	$a = 0.3$ $b = 0.7$	18.467	932.89	30.129
	$a = 0.8$ $b = 0.2$	19.996	661.163	35.683
	$a = 0.7$ $b = 0.3$	19	824.63	32.366
$\mu = 0.5$	$a = 0.2$ $b = 0.8$	18.476	930.87	29.929
	$a = 0.3$ $b = 0.7$	18.823	859.516	30.024
	$a = 0.8$ $b = 0.2$	19.942	664.186	39.233
	$a = 0.7$ $b = 0.3$	19.864	676.456	35.495

Table 2: PSNR, MSE and time elapsed for SAD with various μ, a and b values for removing the object

		PSNR (dB)	MSE	Time elapsed (s)
$\mu = 0.7$	$a = 0.2$ $b = 0.8$	19.148	797.33	57.666
	$a = 0.3$ $b = 0.7$	19.41	750.926	55.902
	$a = 0.8$ $b = 0.2$	19.986	657.69	72.93
	$a = 0.7$ $b = 0.3$	19.461	742.146	65.982
$\mu = 0.5$	$a = 0.2$ $b = 0.8$	19.223	783.92	56.832
	$a = 0.3$ $b = 0.7$	19.139	799.41	57.946
	$a = 0.8$ $b = 0.2$	20.03	650.4	71.197
	$a = 0.7$ $b = 0.3$	19.948	663.356	78.249

5.2. Comparison with existing methods for removing objects

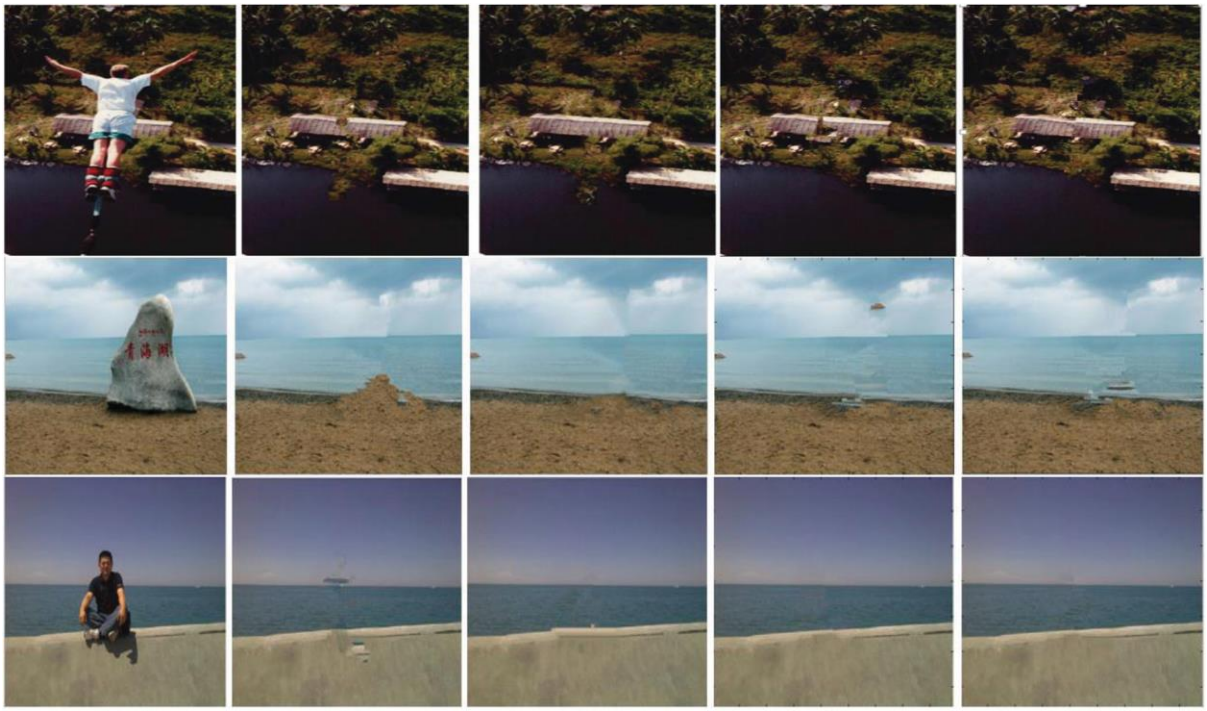


Fig 5.2 Comparison with the existing image inpainting methods for large area removal: column (a) Input image; column (b) Results with Criminisi ; column (c) Results with exemplar based image inpainting; column (d) Results with proposed method for μ = 0.7, a = 0.8 and b = 0.2 using SSE;(e) Results with proposed method for μ = 0.5, a = 0.8 and b = 0.2 using SAD.

5.3 Repairing the damaged images using inpainting

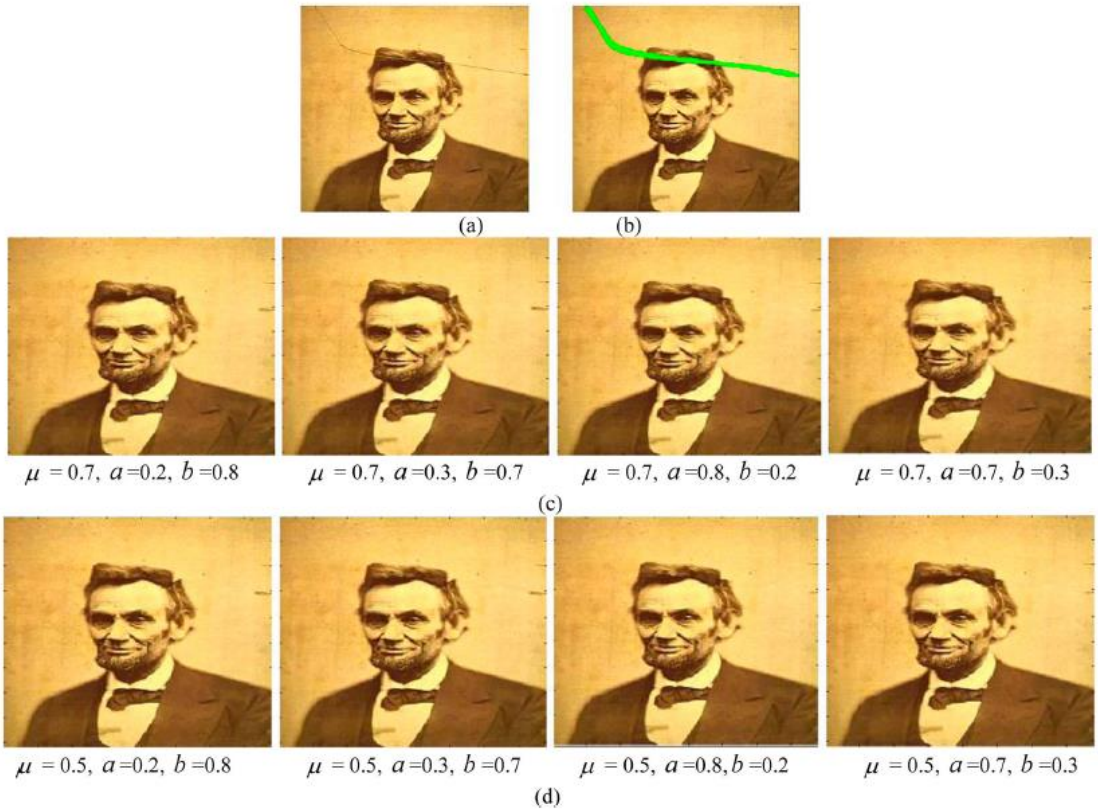


Fig 5.3 Inpainting image results for repairing damaged images using SSE: (a) Damaged image; (b) Mask of damaged portion; (c)inpainted image for μ = 0.7 and four combinations of *a* and *b* values; (d) inpainted image for μ = 0.5 and four combinations of *a* and *b* values.

Table 3: PSNR, MSE and time elapsed for SSE with various μ, a and b values for the repairing image

		PSNR (dB)	MSE	Time elapsed (s)
$\mu = 0.7$	$a = 0.2$ $b = 0.8$	35.345	19.223	76.022
	$a = 0.3$ $b = 0.7$	35.419	18.866	60.097
	$a = 0.8$ $b = 0.2$	35.554	18.476	83.266
	$a = 0.7$ $b = 0.3$	35.597	18.163	80.21
$\mu = 0.5$	$a = 0.2$ $b = 0.8$	35.395	18.973	58.341
	$a = 0.3$ $b = 0.7$	35.318	19.33	67.924
	$a = 0.8$ $b = 0.2$	35.716	17.666	84.188
	$a = 0.7$ $b = 0.3$	35.469	18.68	82.059

Table 4: PSNR, MSE and time elapsed for SAD with various μ, a and b values for the repairing image

		PSNR (dB)	MSE	Time elapsed (s)
$\mu = 0.7$	$a = 0.2$ $b = 0.8$	32.703	35.466	115.959
	$a = 0.3$ $b = 0.7$	33.35	30.566	122.242
	$a = 0.8$ $b = 0.2$	35.544	18.366	188.561
	$a = 0.7$ $b = 0.3$	35.539	18.397	177.924
$\mu = 0.5$	$a = 0.2$ $b = 0.8$	33.129	32.166	125.366
	$a = 0.3$ $b = 0.7$	35.182	19.98	146.49
	$a = 0.8$ $b = 0.2$	35.645	17.95	196.115
	$a = 0.7$ $b = 0.3$	35.63	18	186.917

5.4. Comparison with the existing methods for repairing the images

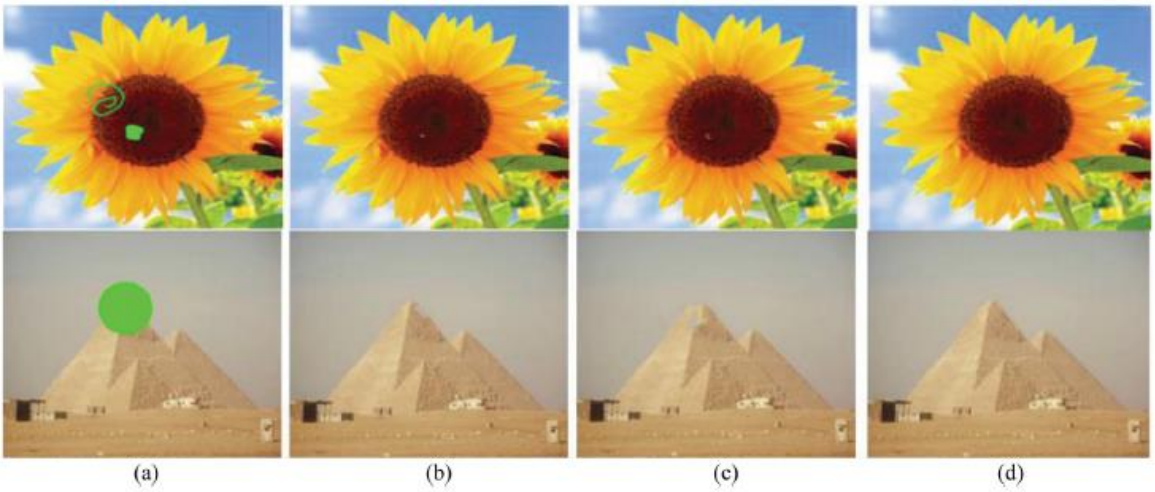


Fig 5.4 Comparison with the existing methods for repairing the damaged images: (a) Mask of damaged portion; (b) Repaired images using Criminisi's method; (c) Repaired images using CBD method; (d) Repaired images using spectral consistency method

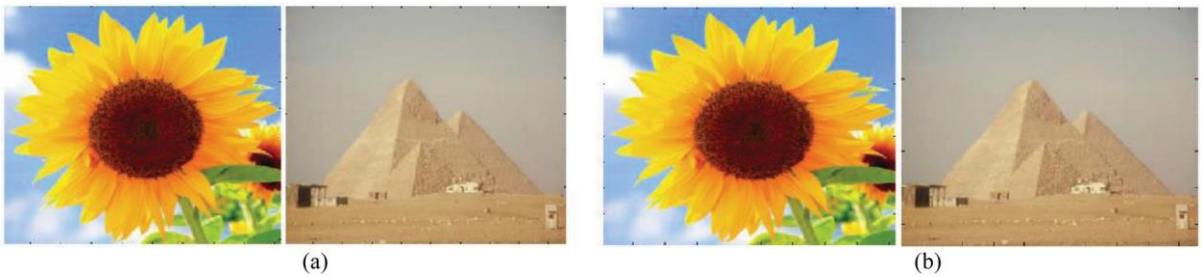


Fig 6.5 Comparison with the existing methods: (a) Results with proposed method for μ = 0.5, a = 0.8 and b = 0.2 using SSE; (b)Results with proposed method for μ = 0.5, a = 0.8 and b = 0.2 using SAD

6. CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

In the report we have proposed an improvement of exemplar-based image inpainting method by analysis the disadvantages of the traditional methods. The proposed method has tried to improve the effect by changing the inpainting priority, the size of the patch and the patch matching method. The experimental result has shown obvious that the improvement increased the accuracy of the patch matching process. As a conclusion, the proposed method has not only performed a better global visual appearance, but also contributed a higher PSNR of the result. Especially for the images with more structure contents and the image which has a large destroyed region. Also in the experiments we have found that only one single scale used in the inpainting is limited. In the future, we will try to use the combination of multi-scale patches to inpaint images.

6.2 Future Scope

There are a few possibilities for further work; implementing the exemplar method on the EVM, optimizing the algorithm for better speed performance, and implementing the exemplar method so that it can inpaint isotopes not just linearly, but in a curvature driven fashion.

The exemplar method is quite powerful. However, it should be adapted to allow more flexible masks; the system should allow specification of what textures grow in where, instead of having all textures grow inward. Thus the algorithm should be adapted so that we have more control over what kinds of textures are grown. The current implementation also looks for textures using a sampling window size of 9x9 (allowing capture of most sized patterns). However, some patterns are larger than 9 pixels across. The algorithm should allow the size of the sampling window to be adjusted, in order to capture these larger patterns. The results generated using the current implementation is quite reasonable.

Time is a big issue. The implementation on Matlab took a few minutes to process a 480 x 640 picture with a mask of approximately 30% of the image. Optimizations should be explored both in the implementation of the algorithms both on the pc and the EVM side. The EVM should be maximally utilized so that as many of the adders, multipliers and registers are occupied at any running time.

The exemplar method has the advantage of finding edges (differing bands of color) and growing these areas first, preserving the edges linearly, respecting the geometry of the objects in the picture. However, it only grows the edges linearly. There are variational approaches that are curvature driven, meaning the diffusion process looks at the way edges are curving and inpainting appropriately. Such a method should be explored for the exemplar case so that it grows much more accurately.

MATLAB CODE

BEST EXEMPLAR HELPER SAD.C

```
/**
 * A best exemplar finder.  Scans over the entire image (using a
 * sliding window) and finds the exemplar which minimizes the sum
 * squared error (SSE) over the to-be-filled pixels in the target
 * patch.
 *
#include "mex.h"
#include <limits.h>

void bestexemplarhelperSAD(const int mm, const int nn, const int m, const int n,
          const double *img, const double *Ip,
          const mxLogical *toFill, const mxLogical *sourceRegion,
          double *best)
{
  register int i,j,ii,jj,ii2,jj2,M,N,I,J,ndx,ndx2,mn=m*n,mmnn=mm*nn;
  double patchErr=0.0,err=0.0,bestErr=1000000000.0;

  /* foreach patch */
  N=nn-n+1;  M=mm-m+1;
  for (j=1; j<=N; ++j) {
    J=j+n-1;
    for (i=1; i<=M; ++i) {
      I=i+m-1;
      /*** Calculate patch error ***/
      /* foreach pixel in the current patch */
      for (jj=j,jj2=1; jj<=J; ++jj,++jj2) {
    for (ii=i,ii2=1; ii<=I; ++ii,++ii2) {
      ndx=ii-1+mm*(jj-1);
      if (!sourceRegion[ndx])
        goto skipPatch;
      ndx2=ii2-1+m*(jj2-1);
```

```
      if (!toFill[ndx2]) {
        err=abs(img[ndx      ] - Ip[ndx2     ]); patchErr += err;
        err=abs(img[ndx+=mmnn] - Ip[ndx2+=mn]); patchErr += err;
        err=abs(img[ndx+=mmnn] - Ip[ndx2+=mn]); patchErr += err;


      }
    }
      }

      /*** Update ***/
      if (patchErr < bestErr) {
    bestErr = patchErr;
    best[0] = i; best[1] = I;
    best[2] = j; best[3] = J;
      }
      /*** Reset ***/
    skipPatch:
      patchErr = 0.0;
    }
  }
}

/* best = bestexemplarhelper(mm,nn,m,n,img,Ip,toFill,sourceRegion); */
void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray *prhs[])
{
  int mm,nn,m,n;
  double *img,*Ip,*best;
  mxLogical *toFill,*sourceRegion;

  /* Extract the inputs */
  mm = (int)mxGetScalar(prhs[0]);
  nn = (int)mxGetScalar(prhs[1]);
  m  = (int)mxGetScalar(prhs[2]);
```

```
  n  = (int)mxGetScalar(prhs[3]);
  img = mxGetPr(prhs[4]);
  Ip  = mxGetPr(prhs[5]);
  toFill = mxGetLogicals(prhs[6]);
  sourceRegion = mxGetLogicals(prhs[7]);

  /* Setup the output */
  plhs[0] = mxCreateDoubleMatrix(4,1,mxREAL);
  best = mxGetPr(plhs[0]);
  best[0]=best[1]=best[2]=best[3]=0.0;

  /* Do the actual work */
  bestexemplarhelperSAD(mm,nn,m,n,img,Ip,toFill,sourceRegion,best);
}
```

INPAINTMODSAD.M

```
function [inpaintedImg,origImg,fillImg,C,D,fillMovie] =
inpaintmodSAD(imgFilename,fillFilename,fillColor)
tic;
%INPAINT  Exemplar-based inpainting.
%
% Usage:   [inpaintedImg,origImg,fillImg,C,D,fillMovie] ...
%              = inpaint(imgFilename,fillFilename,fillColor)
% Inputs:
%   imgFilename    Filename of the original image.
%   fillFilename   Filename of the image specifying the fill region.
%   fillColor      1x3 RGB vector specifying the color used to specify
%                  the fill region.
% Outputs:
```

```
%   inpaintedImg   The inpainted image; an MxNx3 matrix of doubles.
%   origImg        The original image; an MxNx3 matrix of doubles.
%   fillImg        The fill region image; an MxNx3 matrix of doubles.
%   C              MxN matrix of confidence values accumulated over all iterations.
%   D              MxN matrix of data term values accumulated over all iterations.
%   fillMovie      A Matlab movie struct depicting the fill region over time.
%
% Example:
%   [i1,i2,i3,c,d,mov] = inpaint('bungee0.png','bungee1.png',[0 255 0]);
%   plotall;          % quick and dirty plotting script
%   close; movie(mov); % grab some popcorn
%
%   author: Sooraj Bhat
%   Modified by Marcel Davey & John Gu on
%   11/30/05 to run on Matlab 7.0.4.365 (R14) Service Pack 2


warning off MATLAB:divideByZero
[img,fillImg,fillRegion] = loadimgs(imgFilename,fillFilename,fillColor);
img = double(img);
origImg = img;
ind = img2ind(img);
sz = [size(img,1) size(img,2)];
sourceRegion = ~fillRegion;

% Initialize isophote values
[Ix(:,:,3) Iy(:,:,3)] = gradient(img(:,:,3));
[Ix(:,:,2) Iy(:,:,2)] = gradient(img(:,:,2));
[Ix(:,:,1) Iy(:,:,1)] = gradient(img(:,:,1));
Ix = sum(Ix,3)/(3*255); Iy = sum(Iy,3)/(3*255);
temp = Ix; Ix = -Iy; Iy = temp;  % Rotate gradient 90 degrees

% Initialize confidence and data terms
C = double(sourceRegion);
```

```
D = repmat(-.1,sz);
iter = 1;
% Visualization stuff
if nargout==6
  fillMovie(1).cdata=uint8(img);
  fillMovie(1).colormap=[];
  origImg(1,1,:) = fillColor;
  iter = 2;
end

% Seed 'rand' for reproducible results (good for testing)
rand('state',0);

% Loop until entire fill region has been covered
while any(fillRegion(:))
  % Find contour & normalized gradients of fill region
  fillRegionD = double(fillRegion); % Marcel 11/30/05
  dR = find(conv2(fillRegionD,[1,1,1;1,-8,1;1,1,1],'same')>0); % Marcel 11/30/05
 %dR = find(conv2(fillRegion,[1,1,1;1,-8,1;1,1,1],'same')>0);  % Original

  [Nx,Ny] = gradient(double(~fillRegion)); % Marcel 11/30/05
 %[Nx,Ny] = gradient(~fillRegion);         % Original
  N = [Nx(dR(:)) Ny(dR(:))];
  N = normr(N);
  N(~isfinite(N))=0; % handle NaN and Inf

  % Compute confidences along the fill front
  for k=dR'
    Hp = getpatch(sz,k);
    q = Hp(~(fillRegion(Hp)));
    C(k) = sum(C(q))/numel(Hp);
 %   C(k)=log2(C(k)+1);
  end
```

```
% Compute patch priorities = confidence term * data term
D(dR) = abs(Ix(dR).*N(:,1)+Iy(dR).*N(:,2)) + 0.001;
% Regularization factor W=0.7
W=0.5;
RC = (1-W)*C(dR)+W;
a=0.8;b=0.2;
priorities = a*RC + b*D(dR);

% Find patch with maximum priority, Hp
[unused,ndx] = max(priorities(:));
p = dR(ndx(1));
[Hp,rows,cols] = getpatch(sz,p);
toFill = fillRegion(Hp);

% Find exemplar that minimizes error, Hq
Hq = bestexemplar(img,img(rows,cols,:),toFill',sourceRegion);

% Update fill region
toFill = logical(toFill);
fillRegion(Hp(toFill)) = false;

% Propagate confidence & isophote values
C(Hp(toFill))  = C(p);
Ix(Hp(toFill)) = Ix(Hq(toFill));
Iy(Hp(toFill)) = Iy(Hq(toFill));

% Copy image data from Hq to Hp
ind(Hp(toFill)) = ind(Hq(toFill));
img(rows,cols,:) = ind2img(ind(rows,cols),origImg);

% Visualization stuff
if nargout==6
  ind2 = ind;
  ind2(logical(fillRegion)) = 1;          % Marcel 11/30/05
```

```
    %ind2(fillRegion) = 1;                % Original
    fillMovie(iter).cdata=uint8(ind2img(ind2,origImg));
    fillMovie(iter).colormap=[];
  end
  iter = iter+1;
end

inpaintedImg=img;


%-------------------------------------------------------------------
% Scans over the entire image (with a sliding window)
% for the exemplar with the lowest error. Calls a MEX function.
%-------------------------------------------------------------------
function Hq = bestexemplar(img,Ip,toFill,sourceRegion)
m=size(Ip,1); mm=size(img,1); n=size(Ip,2); nn=size(img,2);
best = bestexemplarhelperSAD(mm,nn,m,n,img,Ip,toFill,sourceRegion);
Hq = sub2ndx(best(1):best(2),(best(3):best(4))',mm);


%-------------------------------------------------------------------
% Returns the indices for a 9x9 patch centered at pixel p.
%-------------------------------------------------------------------
function [Hp,rows,cols] = getpatch(sz,p)
% [x,y] = ind2sub(sz,p);  % 2*w+1 == the patch size
w=4; p=p-1; y=floor(p/sz(1))+1; p=rem(p,sz(1)); x=floor(p)+1;
rows = max(x-w,1):min(x+w,sz(1));
cols = (max(y-w,1):min(y+w,sz(2)))';
Hp = sub2ndx(rows,cols,sz(1));


%-------------------------------------------------------------------
% Converts the (rows,cols) subscript-style indices to Matlab index-style
% indices.  Unforunately, 'sub2ind' cannot be used for this.
```

```
%--------------------------------------------------------------------
function N = sub2ndx(rows,cols,nTotalRows)
X = rows(ones(length(cols),1),:);
Y = cols(:,ones(1,length(rows)));
N = X+(Y-1)*nTotalRows;


%--------------------------------------------------------------------
% Converts an indexed image into an RGB image, using 'img' as a colormap
%--------------------------------------------------------------------
function img2 = ind2img(ind,img)
for i=3:-1:1, temp=img(:,:,i); img2(:,:,i)=temp(ind); end;


%--------------------------------------------------------------------
% Converts an RGB image into a indexed image, using the image itself as
% the colormap.
%--------------------------------------------------------------------
function ind = img2ind(img)
s=size(img); ind=reshape(1:s(1)*s(2),s(1),s(2));


%--------------------------------------------------------------------
% Loads the an image and it's fill region, using 'fillColor' as a marker
% value for knowing which pixels are to be filled.
%--------------------------------------------------------------------
function [img,fillImg,fillRegion] = loadimgs(imgFilename,fillFilename,fillColor);
img = imread('D:\harsha mp\sagar\in3.png');
 %mask= imread('D:\harsha mp\sagar\ma7.png');
 %fillImg=imresize(mask,[443,329]);
fillImg = imread('D:\harsha mp\sagar\ma3.png');
fillRegion = fillImg(:,:,1)==fillColor(1) & ...
    fillImg(:,:,2)==fillColor(2) & fillImg(:,:,3)==fillColor(3);
% fillRegion = fillImg(:,:,1)==fillColor(1) & ...
```

```
%     fillImg(:,:,2)==fillColor(2) & fillImg(:,:,3)==fillColor(3);

function [A] = normr(N)
   for ii=1:size(N,1)
      A(ii,:) = N(ii,:)/norm(N(ii,:));
   end
   toc;
```

MYMASK.M

```
% Demo to have the user freehand draw an irregular shape over a gray scale image.
% Then it creates new image with a specified color inside the drawn shape.
% clc;  % Clear command window.
% clear;    % Delete all variables.
% close all;    % Close all figure windows except those created by imtool.
% imtool close all; % Close all figure windows created by imtool.
workspace;  % Make sure the workspace panel is showing.
fontSize = 16;
% Give the path of image which is part of image is to be masked
grayImage = imread('D:\harsha mp\sagar\in3.png');
[rows, columns, numberOfColorChannels] = size(grayImage);
imshow(grayImage, []);
axis on;
title(' freehand mark Image', 'FontSize', fontSize);
set(gcf, 'Position', get(0,'Screensize')); % Maximize figure.
message = sprintf('Left click and hold to begin drawing.\nSimply lift the mouse button to
finish');
uiwait(msgbox(message));
hFH = imfreehand();
% Create a binary image ("mask") from the ROI object.
binaryImage = hFH.createMask();
xy = hFH.getPosition;
```

```
%--------------------------------------------------------------------------------------
% Now make it smaller so we can show more images.
% subplot(2, 2, 1);
figure,imshow(grayImage);
axis on;
drawnow;
title('Original Image', 'FontSize', fontSize);

% Display the freehand mask.
% subplot(2, 2, 2);
figure,imshow(binaryImage);
axis on;
title('source region=0,Fill region=1', 'FontSize', fontSize);

% If it's grayscale, convert to color
if numberOfColorChannels < 3
    rgbImage = cat(3, grayImage, grayImage, grayImage);
else
    % It's really an RGB image already.
    rgbImage = grayImage;
end
% Extract the individual red, green, and blue color channels.
redChannel = rgbImage(:, :, 1);
greenChannel = rgbImage(:, :, 2);
blueChannel = rgbImage(:, :, 3);

% Specify the color we want to make this area.
desiredColor = [0, 255,0]; % Purple
% Make the red channel that color
redChannel(binaryImage) = desiredColor(1);
greenChannel(binaryImage) = desiredColor(2);
blueChannel(binaryImage) = desiredColor(3);
% Recombine separate color channels into a single, true color RGB image.
rgbImage = cat(3, redChannel, greenChannel, blueChannel);
```

```
% Display the image.
% subplot(2, 2, 3);
 figure,imshow(rgbImage);
 title('Filling Region Indicated with color', 'FontSize', fontSize);
%-------------------------------------------------
 %-------------------------------------------------
grayImage =rgbImage ;
[rows, columns, numberOfColorChannels] = size(grayImage);
imshow(grayImage, []);
axis on;
title(' freehand mark Image', 'FontSize', fontSize);
set(gcf, 'Position', get(0,'Screensize')); % Maximize figure.
message = sprintf('Left click and hold to begin drawing.\nSimply lift the mouse button to
finish');
uiwait(msgbox(message));
hFH = imfreehand();
% Create a binary image ("mask") from the ROI object.
binaryImage = hFH.createMask();
xy = hFH.getPosition;
%-----------------------------------------------------------------------------------------
% Now make it smaller so we can show more images.
% subplot(2, 2, 1);
figure,imshow(grayImage);
axis on;
drawnow;
title('Original Image', 'FontSize', fontSize);

% Display the freehand mask.
% subplot(2, 2, 2);
figure,imshow(binaryImage);
axis on;
title('source region=0,Fill region=1', 'FontSize', fontSize);

% If it's grayscale, convert to color
```

```
if numberOfColorChannels < 3
    rgbImage = cat(3, grayImage, grayImage, grayImage);
else
    % It's really an RGB image already.
    rgbImage = grayImage;
end
% Extract the individual red, green, and blue color channels.
redChannel = rgbImage(:, :, 1);
greenChannel = rgbImage(:, :, 2);
blueChannel = rgbImage(:, :, 3);

% Specify the color we want to make this area.
desiredColor = [0, 255,0]; % Purple
% Make the red channel that color
redChannel(binaryImage) = desiredColor(1);
greenChannel(binaryImage) = desiredColor(2);
blueChannel(binaryImage) = desiredColor(3);
% Recombine separate color channels into a single, true color RGB image.
rgbImage = cat(3, redChannel, greenChannel, blueChannel);
% Display the image.
% subplot(2, 2, 3);
 figure,imshow(rgbImage);
 title('Filling Region Indicated with color', 'FontSize', fontSize);
```

PLOTALL.M

```
% This is a simple script to plot some diagrams.
% It assumes you have already run inpaint, storing into the
% variables i1,i2,i3,c,d, like so:
%
% [i1,i2,i3,c,d]=inpaint('bw0.png','bw2.png',[0 255 0]);
```

```
figure;image(uint8(i2)); title('Original image');
figure;image(uint8(i3)); title('Fill region');
figure;image(uint8(i1)); title('Inpainted image');
figure;imagesc(c); title('Confidence term');
figure;imagesc(d); title('Data term');

% figure;
% subplot(121);imagesc(c); title('Confidence term');
% subplot(122);imagesc(d); title('Data term');
% Matlab Code for PSNR and MSE

InputImage=i1;
ReconstructedImage=i2;
n=size(InputImage);
 M=n(1);
 N=n(2);
 MSE = sum(sum((InputImage-ReconstructedImage).^2))/(M*N);
PSNR = 10*log10(256*256/MSE);
 fprintf('\nMSE: %7.2f ', MSE);
 fprintf('\nPSNR: %9.7f dB', PSNR);
```

REFERENCES

[1] B. Janardhana Rao, Y. Chakrapani & S. Srinivas Kumar (2018): Image Inpainting Method with Improved Patch Priority and Patch Selection, IETE Journal of Education
To link to this article: https://doi.org/10.1080/09747338.2018.1474808

[2] Guillermo Sapiro, "Image Inpainting", SCIAM News, Volume 35, #4

[3] Criminisi A, Perez P, Toyama K (2004) Region filling and object removal by examplar-based image inpainting. IEEE Trans Image Process 33:1200–1212

[4] Ram I, Elad M, Cohen I (2013) Image processing using smooth ordering of its ptches. IEEE Trans Image Process 22(7):2764–2774

[5] Huang H-Y, Hsiao C-N (2010) A patch-based image inpainting based on structure consistence. Computer Symposium (ICS): 165-170

[6] Ou J, Chen W, Pan B, Li Y (2016) A new image inpainting algorithm based on DCT similar patches features. In: Computational Intelligence and Security (CIS): 152-155

[7] Mairal J, Elad M, Sapiro G (2008) Sparse representation for color image restoration. IEEE Trans Image Process 17(1):53–69

[8] Xu Z, Sun J (2010) Image inpainting by patch propagation using patch sparsity. IEEE Trans Image Process 19(5):1153–1165

[8] Ružić T, Pižurica A (2015) Context-aware patch-based image Inpainting using Markov random field modeling. IEEE Trans Image Process 24(1):444–456

[9]Papyan V, EladM (2016) Multi-scale patch based image restoration. IEEE Trans Image Process 25(1):249–261

[10] Wang H (2015) Inpainting of Potala Palace murals based on sparse representation. In: Biomedical Engineering and Informatics (BMEI): 737-741

[11] Munawar A, Creusot C (2015) Structural inpainting of road patches for anomaly detection., 14th IAPR International Conference on Machine Vision Applications (MVA): 41-44